VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

Ladění výkonu GPU pro CUDA

GPU Performance Tuning for CUDA

2011/2012

Boháč Petr

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Boháč**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Ladění výkonu GPU pro CUDA
GPU Performance Tuning for CUDA

Zásady pro vypracování:

Dnešní grafické karty firmy nVidia umožňují díky CUDA řešit stejné výpočetní problémy jako na běžném procesoru. Souhrnný výpočetní výkon GPU pak závisí na počtu výpočetních jednotek, šířce paměťové sběrnice, rychlosti celého jádra, rychlosti výpočetních jednotek a rychlosti paměti. Zatímco počet výpočetních jednotek a šířku sběrnice ovlivnit nemůžeme, jednotlivé rychlosti ovlivnit můžeme pomocí specializovaných programů. S rostoucí rychlostí roste i pravděpodobnost chyby. Na druhou stranu lze chybovost snížit snížením teploty čipu nebo zvýšením napětí.

Student ve své práci provede následující kroky:
1. Výběr vhodné grafické karty.
2. Výběr programů pro ladění frekvencí a testování výkonu.
3. Výběr, případně tvorba programu pro detekci chyb jádra a paměti.
4. Ladění frekvencí a měření chybovosti v závislosti na teplotě.
5. Analýza škálovatelnosti vybraných CUDA aplikací.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Štěpán Šrubař**

Datum zadání: 19.11.2010
Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že tato práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, kterou jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Petr Bohdě, Bohdě

Jméno a příjmení, podpis

Abstrakt

Tématem této práce je ladění parametrů grafické karty nVidia a dopad na výkon ve vybraných CUDA aplikacích. Nejprve popíšu programování na grafické kartě obecně a zaměřím se na CUDA. Následně vyberu vhodnou grafickou kartu nVidia a externí program k ladění výkonu této grafické karty. Zvolím nejrůznější testovací aplikace od úpravy audia, videa, změny formátu souboru až po kryptografické aplikace. V tabulkách a grafech porovnávám výkon aplikace při různých nastaveních parametrů grafické karty. Dále měřím chybovost jádra grafické karty v závislosti na teplotě, ale i frekvencích. Navíc ukážu rozdíly ve spotřebě v různých nastaveních.

Abstract

The topic in this work is tuning of parameters graphic card nVidia and impact to performance on chosen CUDA applications. First I will describe programming on graphic card in common and after that I focus on CUDA. Afterwards I will choose appropriate graphic card nVidia and external program to tuning performance of this graphic card. I will choose various test applications from editing of audio, video, format change of file to cryptographic application. In tables and graphs I will compare performance of applications at different setting of parameters on graphic card. After that I measure error rate of graphic card in dependence on temperature and frequency. In addition I will show difference in consumption for various setting of parameters.

Klíčová slova

CUDA, nVidia, GPGPU, MSI Afterburner, ladění výkonu

Keywords

CUDA, nVidia, GPGPU, MSI Afterburner, performance tuning

Obsah

1. Úvod

Herní průmysl je v současné době vysoce výdělečné odvětví. Existuje spousta špičkových vydavatelů her a ti se předhání v kvalitě svých produktů, aby byli pravě oni ti nejúspěšnější. Vytváří tedy stále dokonalejší hry se zaměřením na špičkovou vizuální stránku. To v dnešní době znamená, že obsahuje textury ve vysokém rozlišení, obrovský rozhled, detailní provedení prostředí a v neposlední řadě implementace fyzikálních zákonů. To klade obrovské nároky na výkon grafické karty. Zde však vstupují na scénu tvůrci grafických karet, kteří ženou své odvětví velkými kroky kupředu. Každým rokem vytváří nové grafické karty s desítky procent výkonu navíc. To dospělo do současného stavu, kdy výpočetní výkon grafické karty předčil mnohonásobně výkon procesoru (central processing unit - CPU). Začalo se uvažovat, jak dále využít obrovský výkon grafické karty, který donedávna sloužil pouze k výpočtu vizuálních scén.

Tak vznikl nápad využít GPU k matematickým výpočtům. Výhodou je i využití paralelismu (výpočet lze provádět současně nad celým proudem dat). Problematikou víceúčelového výpočtů na grafické kartě se zabývá skupina GPGPU (General-Purpose computation on GPUs), jejichž webová adresa je http://www.gpgpu.org/

Při koupi nové grafické karty je důležité najít srovnání výkonu současných produktů. Jen tak zajistíme, že si zvolíme nejideálnější kartu, za cenu, kterou jsme ještě ochotni investovat. V tu chvíli má smysl měřit výkon, spolehlivost a zvyšovat parametry grafické karty, aby byl dosažen co nejvyšší výkon, bez nutnosti investovat další peníze ke koupi lepší grafické karty.

Cíle této práce je tedy objasnění výhod využití matematických výpočtů na grafické karty. Dále škálovatelnost GPU s různými parametry a zjišťování chybovosti při různých zatíženích a teplotách GPU a měření příkonu grafické karty v závislosti na zvyšování frekvencí a napětí.

2. GPGPU

2.1 Co je GPGPU

Jak už bylo v úvodu naznačeno, výkon grafických karet (dále už jen GPU) mnohonásobně předčil výkon klasických procesorů a byla snaha toho využit i někde jinde, než ve vykreslování grafických scén. Tak vznikl v roce 2006 nápad pro matematické výpočty na GPU. Tyto výpočty se označují pod zkratkou GPGPU[9] (General-Purpose computation on GPUs, v českém překladu Víceúčelové výpočty na GPU). Zde se vývoj ubírá jiným směrem než výpočty na CPU, kde se využívá malý počet jader (v současnosti až 8 jader), které jsou však taktovány na vysoké frekvence (až 4 GHz). Protože vývoj dospěl do stádia, kde GPU mají díky snaze o co nejvyšší výkon v oblasti vykreslování grafiky v současnosti až 512 jader a jsou taktovány sice na nižší frekvence (asi 900 MHz), hojně se využívá paralelizace ve výpočtech. Tyto jádra jsou však jinak specializovaná (má méně vnitřní logiky), dokážou zvládat pouze jednoduché instrukce, ale obrovskou rychlostí a s využitím mnohonásobné paralelizace.

2.1.1 Oblast využití pro GPGPU

Obrovská výpočetní síla grafických procesorů je způsobena úzce specializovanou architekturou, která je už spoustu let vyvíjena k dosažení maximálně možného výkonu u paralelních úloh, souvisejících s vykreslováním grafických scén. Díky stále rostoucímu zájmu z řad vývojářů o obecné výpočty na GPU a snahou tvůrců grafických karet vytvořit co možná nejuniverzálnější výpočetní jednotky dnes umožňuje užití GPU i pro aplikace, které nemají co dočinění s vykreslováním grafiky. Existuji však spousta programů, které v současnosti nejsou vhodné k výpočtům na GPU. Příkladem může být například práce s textem nebo kryptografie (tento problém se zdá být minimalizovaný). Jedná se o úlohy které nejde téměř vůbec paralelizovat a také nevyhovuje přesnost výpočtů na GPU.

Na grafických procesorech dlouhá léta chyběla podpora pro práci s celými čísly. Absence operace pro bitový posun a logických operací. To znemožnilo některé výpočty, jejichž hlavní zástupce je kryptografie. Tento neduh byl naštěstí nedávno odstraněn a byla přidána plná podpora v nejvýznamnějších GPGPU technologiích (CUDA, OpenCL)[4]. GPU nejsou vůbec vhodná u úloh, kde je kladený důraz na přesnost výsledku. To je způsobeno tím, že je podpora pouze pro 32 bitová čísla s plovoucí čárkou (CPU má v tomto ohledu navrch, podporuje i 64 bitová čísla).

Stále však existují překážky pro problémy, které jsou ideální pro výpočty na GPU. Ale toto odvětví má krátkou historii a zatím má stále své limity mimo oblast vykreslování grafiky. To je také hlavní účel grafické karty, GPGPU výpočty jsou zatím stále ve vývoji. Pravidelně však vychází nové grafické karty s novějšími instrukčními sadami a lepšími parametry pro výpočty.

2.1.2 Metody výpočtů

Obecně programování na grafické kartě je svým způsobem podobný práci s proudem dat. Existuje pár základních metod, které se hojně využívají v kombinaci s proudy dat a jsou využity ve většině aplikací, využívající výpočty na GPU. Tyto metody jsou mapa, redukce, filtrace proudu, rozptýlení, shromažďování, řazení, vyhledávání[9].

Mapa (Map)

Jedná se o nejjednodušší metodu pro práci s proudem dat. Principem mapy je aplikace dané funkce na všechny elementy v proudu dat. Jednoduchým příkladem může být násobení každého elementu proudu konstantou. V praxi se může jednat například o zvýšení osvícení scény. Mapa je jednoduchá pro implementaci na GPU. Data se načtou do paměti grafické karty podobně jako textura a funkce se provede pomocí GPU v okamžiku vytváření scény. Výsledný proud dat má stejný počet elementů jako původní proud dat.

Redukce (Reduce)

V některých případech je potřeba, aby se ze vstupního proudu dat vytvořil výstupní proud, který obsahuje menší počet elementů. V extrémním případě proud o jednom elementu. To je možno dokázat pomocí metody redukce. Obecně však musí byt metody využita ve více krocích a to pomocí střídavého načítání a poté zpracování dvou scén. Po každém průchodu se velikost výstupu sníží na polovinu.

Filtrace proudu (Stream Filtering)

Filtrace proudu je v podstatě speciální případ redukce. Ale prvky z proudu dat se mažou pomocí nějakých kriterií. Protože pozice prvků ani jejich počet není předem známý, metoda se musí provádět ve více krocích. V praxi se dosáhnem filtrace pomoci užití více metod výpočtu, následovaných vícenásobným průchodem proudu.

Shromažďování (Gather)

Tato metoda umožňuje načíst textury v náhodných přístupech, takže může shromáždit informace z jakékoliv buňky v mřížce nebo z několika buněk současně.

Řazení (Sort)

Pomocí řazení můžeme z neuspořádaných prvků v proudu dat vytvořit uspořádanou řadu prvků v proudu. Zde není možné použít algoritmy, které se využívají při počítaní na CPU, protože jsou jejich kroky závislé a vyžadují operaci rozptýlení.

Pro většinu základních implementací se využívají řadící sítě (sorting networks). Myšlenkou pro tento přístup je princip řazení, který je založen na pevně stanoveném počtu kroků, který je nezávislý na vstupních datech. Dalším aspektem je, že všechny uzly řadící sítě mají pevné komunikační cesty, problém tedy lze vyřešit pomocí metody shromáždění. Reálně je možné docílit uspořádání proudu v $O(n \log^2(n))$ krocích.

Vyhledávání (Search)

Metoda vyhledávání umožňuje programátorovi nalézt jistý prvek nebo množinu prvků v proudu dat, které odpovídají hledanému prvku. V tomto případě se výkon GPU nepoužívá ke zrychlení výpočtu, ale využívá se paralelizace, aby mohlo běžet více vyhledávání současně. Tím se zvýší propustnost. Jeden ze způsobů realizace je binární vyhledávání, které pracuje nad seřazenými daty a vždy porovnává střední prvek s prvkem, který vyhledáváme. Po zjištění výsledku aplikuje tento postup pro levou nebo pravou půlku dat, dokud nevyhledá potřebná data nebo neověří, že takový prvek neexistuje.

V praxi nelze paralelizovat vyhledávání jednoho prvku, protože binární vyhledávání je sekvenční. Díky GPU však můžeme jednoduše vyhledávat více prvků nad stejnými daty.

2.1.3 Datové struktury

Každá z metod pracuje s daty, která musí být uložena ve struktuře co nejefektivnější pro výpočty na GPU. Taková datová struktura musí zvládnout paralelní přístup k datům a musí podporovat rychlý přístup. Aby bylo možno problém řešit co nejefektivněji na GPU, datová struktura musí respektovat vnitřní paměťový model.

Prvky jsou tedy uloženy výhradně v podobě textury, která má dva rozměry. jednorozměrná data mohou být uložena jako jeden řádek, ale protože maximální velikost textury v jednom rozměru je omezena, využívá se mapování z jednodimenzionálního do dvoudimenzionálního prostoru. V současnosti se nejčastěji využívají řídká pole, hustá pole (statická i dynamická) a adaptivní struktury.

2.1.4 Možnosti využití

V současnosti existuje pár odvětví, pro které se hodí výpočty na GPU, zejména díky přínosu paralelizace. V těchto případech může být aplikace až 150x rychlejší než při běhu na CPU.

Jedná se zejména o diferenciální rovnice, které se využívají ve spoustě odvětví. Navíc se využívá pro simulaci fyzikálních jevů. Což v praxi znamená výpočty pro obrovské množství dat, takový problém se dá přirozeně paralelizovat a může být efektivně realizován na GPU.

GPGPU je přínosem také v oblasti Lineární algebry. Práce s maticemi a vektory lze taky rozdělit do menších nezávislých celků a tedy velice dobře paralelizovat. Jedná se o tak velké zrychlení, že mohou být problémy se šířkou paměťové sběrnice. Avšak každou novou řadou grafických karet jdou parametry vždy kus dopředu.

Své využití si našel i v oblasti distribuovaných výpočtů. Je založen na principu rozdělení problému na dílčí úkoly, které jsou prováděny na domácích počítačích po celém světě a výsledky jsou shromažďovány. Tak docílíme vysokého paralelního výkonu. V roce 2000 vznikl první takový projekt, který však využíval výpočtů na CPU. Na to navázal v roce 2006 první vědecký projekt založený na výpočtech na grafické kartě vyvíjený Standfordskou Univerzitou s názvem Folding@Home[7].

2.2 Hardwarové prostředky GPGPU

2.2.1. Programovatelné jednotky

2.2.1.1 Obecně

Vývoj jde stále kupředu i v oblasti výpočetních jednotek grafických karet. To je způsobeno zejména zvyšování požadavků na tyto jednotky. Na počátků GPGPU výpočtů bylo možno vytvářet velmi krátké programy, které nepodporovaly jakékoliv větvení a využívaly omezené počty registrů. Vlastnosti a tím i schopnost využití programovatelných jednotek na grafických kartách se začaly rozšiřovat, ale cesty rozšíření se lišily mezi nejvýznamnějšími výrobci. Proto bylo potřeba usměrnit tento vývoj jedním směrem a tím pádem normalizovat, aby bylo možno využít vylepšené jednotky v praxi. Pro tento účel byly vytvořeny třídy Shader Model. Tyto třídy určují minimální požadavky u grafického čipu [1]. V tabulce 1 je uvedeno, jak se vylepšovaly vlastnosti Shader Modelů postupem času od verze 1.0 až po 4.0 . Informace jsou čerpány hlavně z vývojové sady DirectX SDK.

Shader Model	1.0 - 1.3	1.4	2.0	3.0	4.0
Max. počet texturovacích instrukcí	4	6 · 2	32	bez limitu	bez limitu
Poziční registr	Ne	Ne	Ne	Ano	Ano
Max. délka programu	8 + 4	8 + 4	32 + 64	≥ 512	≥ 65536
Max. počet vykonaných instrukcí	8 + 4	6·2+8·2	32 + 64	65536	bez limitu
Počet interpolačních registrů	2 + 8	2 + 8	2 + 8	10	32
Predikce instrukcí	Ne	Ne	Ne	Ano	Ne
Registry k indexovaní vstupů	Ne	Ne	Ne	Ano	Ano
Dočasné registry	2	6	12 až 32	32	4096
Registry konstant	8	8	32	224	16 · 4096
Čítač cyklů	Ne	Ne	Ne	Ano	Ano
Dynamická kontrola toku	Ne	Ne	Ne	24	Ano
Operátory bitového posunu	Ne	Ne	Ne	Ne	Ano
Podpora celého čísla	Ne	Ne	Ne	Ne	Ano

Tabulka 1: Srovnání vlastností různých verzí Shader Model

Z tabulky 1. lze vyčíst, že vylepšení jsou opravdu markantní. Například maximální délka programu mohla být v první verzi 12 instrukcí, ve verzi 3.0 už 512 a ve verzi 4.0 už dokonce 65536 instrukcí. Další zajímavostí je zde podpora bitového posunu a celých čísel až od verze 4.0. V současností je nejvyšší standart Shader Model 5.0, který však jen lehce vylepšuje verzi 4.0. Nejvýznamnějším je tzv. Compute Shader, který zvyšuje rychlost výpočtů rozdělením problému do více vláken a lépe sdílí data. Nová verze Shader Model vždy přichází spolu s aplikačním rozhraním DirectX pro operační systém od společnosti Microsoft. Nová řada grafických karet vždy podporuje nejvyšší třídu Shader Model, ať se jedná o model AMD nebo nVidia. V tabulce 2 ukážu příklady různých řad grafických karet a Shader Model, který podporují.

Grafická karta	SM
GeForce GTX 570	SM 5.0
GeForce GTX 470	SM 5.0
Geforce GTX 280	SM 4.1
Radeon HD 7970	SM 5.0
Radeon HD 6970	SM 5.0
Radeon HD 5970	SM 5.0

Tabulka 2: Podpora Shader Model u nejmodernějších grafických karet

Z tabulky je patrné, že podpora Shader Model 5.0 je u grafických karet nVidia už od řady GTX 4XX, což je necelé 3 roky starý produkt. AMD (v době vydání Radeon HD 5XXX se společnost jmenovala ATi) přišla s podporou Shader Model 5.0 už u své řady Radeon HD 5XXX. Tato karta je na trhu 4 roky.

2.2.1.2 Instrukční sady grafických karet

Z důvodu primárního účelu grafických karet se instrukční sady samozřejmě liší od těch klasických na CPU. Ale instrukční sady pro výpočty na GPU se rozšiřují a díky přidání podpory pro větvení programu v Shader Model 3.0, podpory celočíselných operací a bitového posunu v Shader Model 4.0 se možnosti téměř vyrovnaly. Tímto se stalo GPGPU konkurenceschopné vůči výpočtům na CPU a díky využití paralelizace vhodné pro spousty druhů výpočtů. V tabulce 3 jsou vypsány příklady instrukcí a jejich popis [1].

Instrukce	Popis
nop	žádná operace
mov	přesun hodnoty
add	součet
abs	absolutní číslo
max	maximum
sincos	sinus/cosinus
loop	počátek cyklu
label	návěstí
call	volání podprogramu

Tabulka 3: Vybrané instrukce

Tyto instrukce mají podporu Shader Modelu 3.0 a vyšší. Původně většina těchto instrukcí byla standardem pro starší verzi, ta však už není podporována.

2.2.2 CUDA

CUDA (Compute Unified Device Architecture) je hardwarová a softwarová architektura vyvíjena společností nVidia. Proto je tato architektura dostupná pouze na grafických kartách stejné společnosti.

2.2.2.1 nVidia CUDA C, C++

Technologie nVidia CUDA byla primárně navržena pro jazyky C, C++ a Fortran[8]. Hlavním cílem společnosti nVidia bylo vytvořit prostředky pro tvůrce počítačových her, které by jim umožnily provádět složité operace na GPU a tím ulevit vytížení na CPU. V případě úspěchu tohoto řešení by získala nVidia dominantní pozici na trhu s grafickými kartami. Od uvedení tohoto řešení netrvalo dlouho a velká světová studia využili tuto technologii ve svých počítačových hrách. Technologie se začala využívat pro výpočet náročných fyzikálních zákonů. To přineslo do herního průmyslu obrovský zvrat, protože za využití současných prostředků CUDA technologie dokázala zpracovat mnohonásobně složitější operace v oblasti gravitace, destrukce objektů, simulace různých povrchů a tkanin až po simulaci tekoucí vody a reakce vlasů na pohyb hlavy.

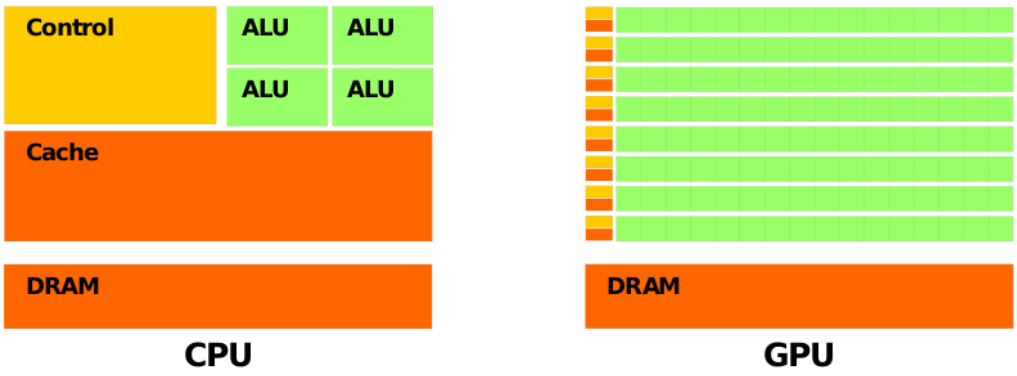
Po výborných výsledcích, které technologie CUDA dosáhla v herním průmyslu v oblastí propracované fyziky u nejmodernějších titulů byla snaha dosáhnou podobného úspěchu i ve vědeckých aplikacích a různých programech, kde by se dal obrovský výkon využít. A tak začaly vznikat první aplikace v oblasti vědy. Jedna z prvních byla nástavba Simulink do oblíbeného programu Matlab, který umožnil použití paralelizace do úloh matematických i fyzikálních. Dále začaly vznikat aplikace pro tvorbu proteinu, které by léčily nevyléčitelné choroby [7] nebo různé aplikace pro převod a úpravu videí.

2.2.2.2 Co je CUDA?

NVIDIA CUDA je technologie, jejíž hlavním účelem je zprostředkovat schopnost paralelních výpočtů na GPU jednotkách grafických karet nVidia (od série GeForce 8XXX výše) a tak umožnit řešení rozsáhlých, komplexních výpočetních problémů v mnohem kratším čase, který by tyto operace trvaly, kdybychom použili klasické CPU. Abychom byli schopni začít používat tuto technologii, je třeba nainstalovat CUDA Engine, který volně poskytuje nVidia a je složen z několika částí.

Součástí CUDA Engine je architektura instrukční sady CUDA (anglická zkratka CUDA ISA) a paralelní výpočetní rozhraní GPU.

Vývojáři na architektuře CUDA v současnosti mohou využít celou řadu programovacích jazyků, avšak nejpopulárnější jsou jazyky C a C++. Avšak podpora je i pro méně využívané programovací jazyky, jako jsou Fortran, Python, Java a OpenCL. Na následujícím obrázku 1 je zobrazení rozdílu mezi architekturou klasických procesoru CPU (do porovnání je zahrnuta i operační paměť Random-Access Memory, častěji známou jako RAM) a grafických karet nVidia schopných výpočtu pomocí technologie CUDA.



Obrázek 1 : Architektury CPU a GPU

Na výše uvedeném obrázku je vidět poměr aritmetickologických jednotek (tzv. ALU), které provádějí samotný výpočet. Oproti standardním procesorům, které dnes v běžné podobě nabízejí nejvíce dvanáct těchto jednotek, grafická karta jich může poskytnout několik stovek - v případě modelu NVIDIA GTX 570, který jsem použil pro tuto práci, je to 480 ALU(CUDA core), extrémním případem je v současné době grafická karta GeForce GTX590, která nabízí uživateli 1024 výpočetních jader - čehož lze využít při výpočtu náročných operací, které lze tzv. paralelizovat, tedy je prováděno více úkonů najednou.

Dalším rozdílem mezi klasickým procesorem typu CPU a procesorem grafické karty typu GPU je to, jakým způsobem přistupují tyto procesory k paměti, ve které jsou uloženy pro výpočet důležité informace, jako jsou mezivýpočty a proměnné potřebné k běhu výpočetního úkonu. K těmto datům pak procesor přistupuje přes tzv. sběrnici, která má ale svá omezení co se týče objemu přenesených dat za jednotku času. Tato schopnost přenést určitý objem dat za jednotku času je nejčastěji uváděna výrobci jako tzv. šířka sběrnice. Operační paměť, kterou používá běžný procesor má v dnešní době šířku sběrnice 64 bitů, zatímco nejmodernější grafické karty mají šířku sběrnice až 768 bitů, z čehož vyplývá, že grafická karta dokáže nejen provést více výpočtů, ale dokáže zároveň mnohem rychleji přistupovat k datům pro tyto výpočty potřebným.

Velkým úskalím grafických karet je však počet instrukcí, které je každé jádro schopné zpracovat a vyšší náročnost na provedení běžných programátorských úkonů.

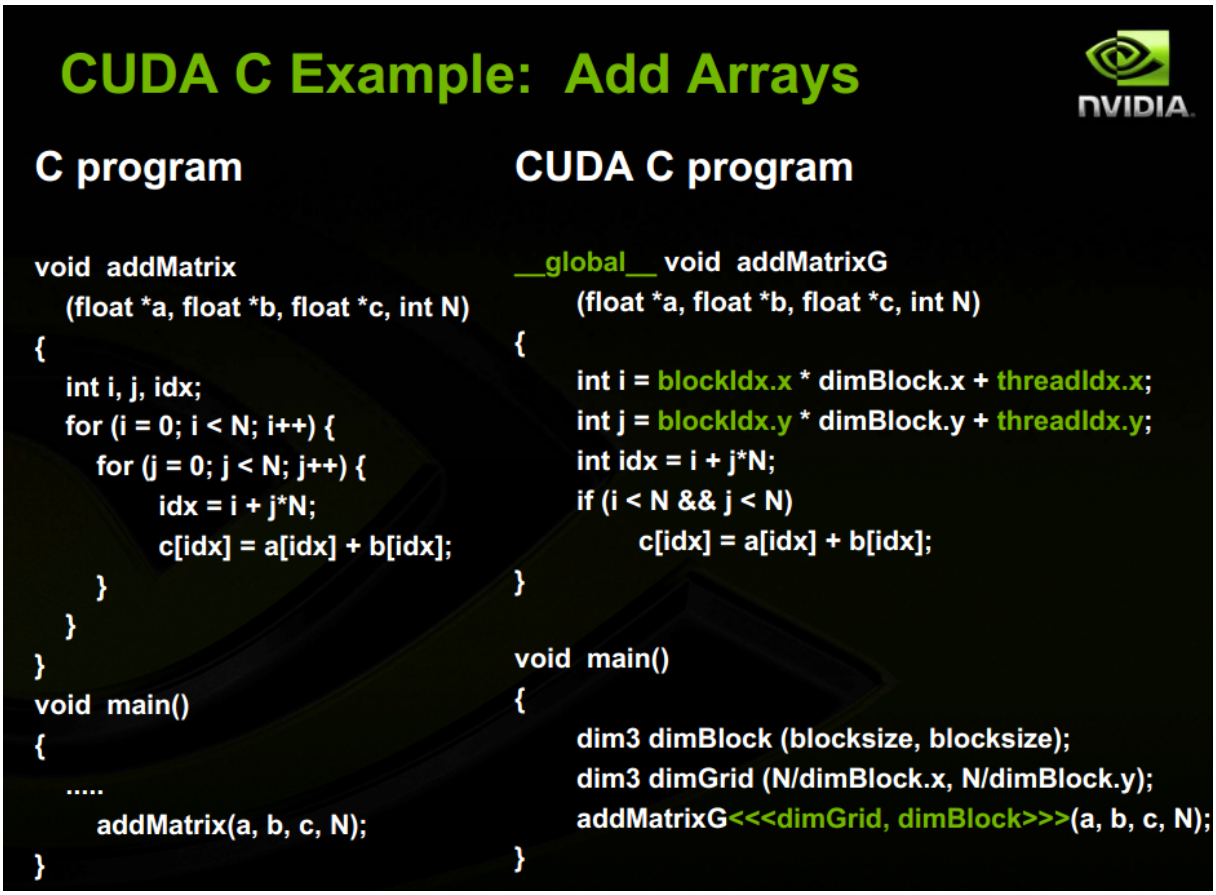
2.3 Softwarové požadavky pro GPGPU

2.3.1 Předpoklady pro programování v CUDA

První nezbytnou součástí je tzv. CUDA ovladač, který nám poskytne nejdůležitější hardwarovou spolupráci grafické karty s knihovnami daného programovacího jazyka[8]. Druhou součástí jsou pak CUDA nástroje a poslední částí pak je CUDA SDK (Software Development Kit) s ukázkami zdrojového kódu. CUDA ovladač poskytne vývojáři základní nástroje pro užití této architektury. Těmito nástroji jsou nvcc což je samotný kompilátor pro jazyk C, CUDA FFT a BLAS knihovny, profiler (pro sledování běhu aplikace a prováděných operací), gdb debugger (pro jednodušší odstraňování chyb) pro GPU, CUDA běhový ovladač a programátorský manuál.

Rozšířením této základní sady je pak CUDA SDK, které obsahuje mnohem více užitečných materiálů ve formě ukázek zdrojového kódu. Mezi těmito ukázkami jsou dema paralelního řazení, násobení matic, transpozice matic, ladění výkonu a užívání časovačů, prefixové sčítání velkých polí a spoustu dalších ukázek[8].

2.3.1.1 Ukázka programu



Obrázek 2. Ukázkový program matice

Z ukázkového příkladu jde vyčíst, že CUDA C je opravdu založené na klasickém programovacím jazyku C, rozšíření se týká hlavně mapování do paralelních vláken [2].

Nejprve je potřeba deklarovat funkci *addMatrix*. Zde jsou programy téměř totožné, v CUDA C přibývá pouze klíčové slovo *_global_* . Ale už při deklaraci vnitřních proměnných *i* a *j* si lze všimnout využití paralelismu. Proměnné jsou vytvořeny jako samostatné vlákna. Samotné algoritmy pro přidání matice jsou si velice podobné. Další rozdíl nastane až ve volání funkce *addMatrix* v metodě *main*. První dvě tvrzení inicializuji dimensionální parametry. Tedy počty vláken a bloků. Přibylo ještě tvrzení *<<<dimGrid, dimBlock>>>* , což je povinná část pro rozběhnutí programu.

2.3.1.2 PhysX

PhysX je engine , simulující fyzikální zákon v reálném čase [3]. Je založen na architektuře CUDA, takže výpočty probíhají na grafické kartě nVidia řady GeForce 8800 a vyšší. PhysX dokáže akcelerovat kapaliny, látky, celé postavy, destrukci objektů a mnohé další. To je umožněno pomocí vysokého výkonu, který zajišťuje paralelizace a jedná se o odvětví ideální pro paralelní výpočty, protože se pracuje s mnoha body současně.

2.3.2 Technologie OpenCL

Jedná se o nejuniverzálnější nástroj, který lze použit na většině operačních systémech jako jsou Microsoft Windows, hlavní distribuce Linuxu (Ubuntu, Fedora, OpenSUSE) a Apple MacOS X [4]. Podpora je i ze strany ovladačů grafických karet. Vývoj OpenCL má na svědomí průmyslové konsorcium Khronos, které jej uveřejnilo v roce 2008.

OpenCL definuje abstraktní hardwarové zařízení ovládané příslušným softwarovým rozhraním, díky tomu aplikace může přistupovat k dostupným výpočetním možnostem na různých platformách.

2.3.3 Technologie DirectCompute

Rozhraní DirectCompute vyvíjí společnost Microsoft [5]. GPGPU výpočty podporuje pouze na operačních systémech této společnosti, kterými jsou Windows Vista a Windows 7. DirectCompute je vyvíjená jako součást Microsoft DirectX kolekce programovatelných rozhraní. Možnosti této technologie jsou velice podobné CUDA a OpenCL.

3. Grafická karta a využívaný software

3.1 Výběr grafické karty k testování

Parametry a specifikace:
Jádro:
NVIDIA GeForce GTX 570 (GF110)
Počet stream procesorů: 480
Výrobní technologie: 40 nm
Paměť:
Kapacita: 1280 MB
Typ: GDDR5
Sběrnice: 320bitová
Propustnost: 160 GB/s
Frekvence:
Jádro: 742 MHz
Shadery: 1484 MHz
Paměť: 4000 MHz
Rozhraní:
PCI Express x16 2.0

Tabulka 4: Parametry grafické karty

Obrázek 3: Asus GTX 570

Technologie:
Microsoft DirectX 11
Shader Model 5.0
Open GL 4.1
NVIDIA PhysX
NVIDIA CUDA

3.1.1 Obecně

K testování jsem zvolil nereferenční (co se týče chlazení, použití kvalitnějších součástek i návrhu PCB, neboli desky plošných spojů) model od společnosti ASUS. Konkrétně se jedná o model ENGTX570 DCII. Jde o následovníka první generace Fermi architektury (vylepšená architektura se označuje pod kódovým jménem nějakého známého vědce, dalším bude Kepler), což byly grafické karty řady GTX 4xx. Tato řada měla problémy s přehříváním grafického jádra, z důvody obrovské spotřeby celého GPU. Tyto problémy byly vyřešeny a vznikla velmi úspěšná a výkonná řada GTX 5xx.

3.1.2 Parametry

Grafická karta je postavená na jádru GF110. Což je stejný čip, jako nejvýkonnější jednočipové GPU GTX 580, jen je zamčen jeden blok CUDA jader (ALU). Má jich tedy 480. Grafická karta je vyrobena 40nm výrobním procesem. Paměti GDDR5 o kapacitě 1280 MB jsou připojeny na 320 bitovou sběrnici. Propustnost paměti tedy činí 160 GB/s. Frekvence jádra jsou oproti referenční hodnotě o trochu zvýšená z 732 MHz na 742 MHz, ale u testování aplikací uvidíme, že to není zdaleka limitní hodnota tohoto čipu. Frekvence shaderů je dvojnásobek frekvence jádra, ale oproti minulé generaci Fermi, na niž je tento čip postaven, frekvence shaderů nejde samostatně měnit, vůči jádru. Tím že tato grafika obsahuje paměti GDDR5, které jsou přímo dělané na použití v grafických kartách dokáže přenést 4 bity za 1 takt, tak při reálné frekvenci paměti 1000 MHz získáme efektivní frekvenci paměti 4000 MHz. Karta samozřejmě obsahuje úsporná opatření, aby v době jejího malého vytížení odebírala co nejméně. To je zajištěno snížením frekvencí na jádru, shaderech a paměti. Díky snížení frekvencí si může karta dovolit snížit i napětí jádra.

3.2 Programy k ladění výkonu GPU

3.2.1 Problematika ladění výkonu

Zájem veřejnosti v této oblasti stále roste. Je tu způsobeno zejména větší dostupností kvalitních ladících programů, které dokážou jednoduše zvyšovat parametry grafické karty. Ale zvyšování výkonu přináší i negativní vlivy. Zvýšení frekvencí způsobuje větší zahřívání grafického čipu, navíc pokud chceme zvýšit výrazně frekvence, tento krok si vyžádá také zvýšení napětí grafického čipu. Dosažením vysokých teplot navíc riskujeme zvýšení chybovosti. Tyto předešlé kroky vedou ke zvýšení spotřeby grafické karty. Ladění výkonu je ošemetný obor. Vystavujeme se dobrovolně riziku zkrácení životního cyklu grafické karty, která v nejhorším případě může skončit zničením. To jsou také hlavní důvody, proč se ve větší míře nevyužívá ladění výkonu v profesionální svéře.

U profesionálních výpočtů není prostor pro chybu, takže pokud se nějaká společnost vydá touto cestou, je třeba zajistit kvalitní chlazení grafické karty. A nejlépe využívat takové výpočetní stroje v klimatizovaných místnostech.

Existuje několik dalších způsobů, jak umožnit zvýšení grafického výkonu. Pořízení nereferenčního návrhu grafické karty případně využití lepšího chlazení (v nejlepším případě vodní chlazení).

Extrémním způsobem pro dosažení maximálního grafického výkonu je využití technologie SLi (od společnosti nVidia) nebo Crossfire (AMD). Tato možnost však vyžaduje pořízení další totožné grafické karty. Tímto způsobem je možno spárovat až čtveřici stejných karet.

Program (první svého druhu) pro ladění výkonu grafické karty nVidia byl Rivatuner. Do té doby byly jen omezené možnosti zvyšování parametrů pomocí ovladačů grafické karty.

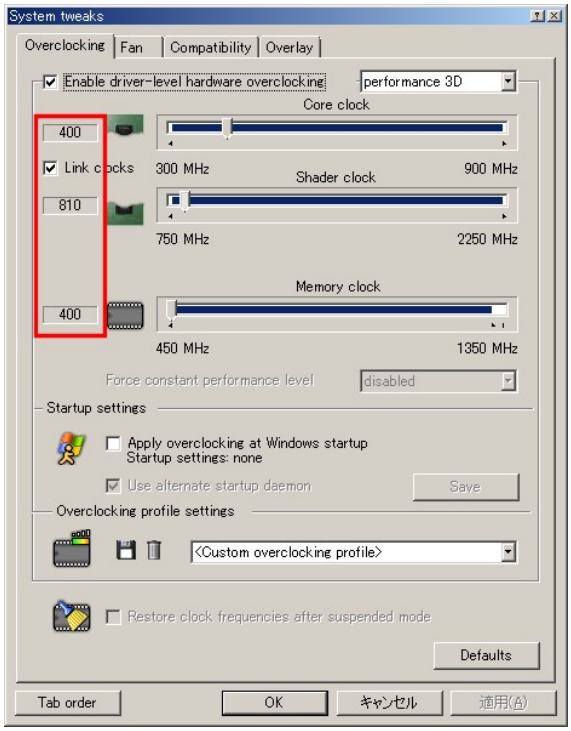
3.2.2 Rivatuner, aneb historie ladění výkonu

Je vytvořen jedním nadšencem z Ruska. Jmenuje se Alexey Nicolaychuk. Tento programátor nebyl spokojen s tím, co dokážou ovladače grafické karty nastavit, tak se pustil do vytvoření vlastního programu. Bylo to v letech 1998, kdy byly na scéně grafické karty Riva TNT od firmy nVidia [6]. V té době se program stal obrovským hitem pro nadšence v ladění výkonu své grafické karty. Postupně získal podporu pro většinu grafických karet na trhu. Nejen že uměl měnit frekvence pamětí a jádra, ale navíc u některých modelů dokázal odemknout softwarově zamknuté bloky grafické karty. Výrobci tak upravují grafické jádra, pokud chtějí z nejvýkonnějšího modelu udělat model střední třídy. Nebo tímto způsobem zamykají ve výrobě nefunkční bloky.

Původně byl Rivatuner program, který měnil hodnoty registru. Poté byl vylepšen a pracoval ve dvou režimech. Dokázal ladit grafickou kartu na úrovni ovladačů nebo na nižší úrovni(low level).

Na úrovni ovladačů Rivatuner upravoval nastavení pomocí registrů a volal funkce ovladačů za účelem změny parametru (zvýšení frekvence jádra atd.). V tomto režimu taky dokázal vyčíst některé informace o grafické kartě (frekvence paměti a jádra, informace o AGP nastavení)

Když se program spustil v režimu nižší úrovně, pracoval přímo s grafickým hardwarem. Pokud se měnily parametry v tomto módu, obcházely se funkce ovladačů a parametry se přímo zapisovaly do registrů na grafické kartě. To však přineslo i jistá rizika a hrozilo poškození grafické karty.



Obrázek 4: Rivatuner a úprava frekvencí

3.2.3 MSI Afterburner

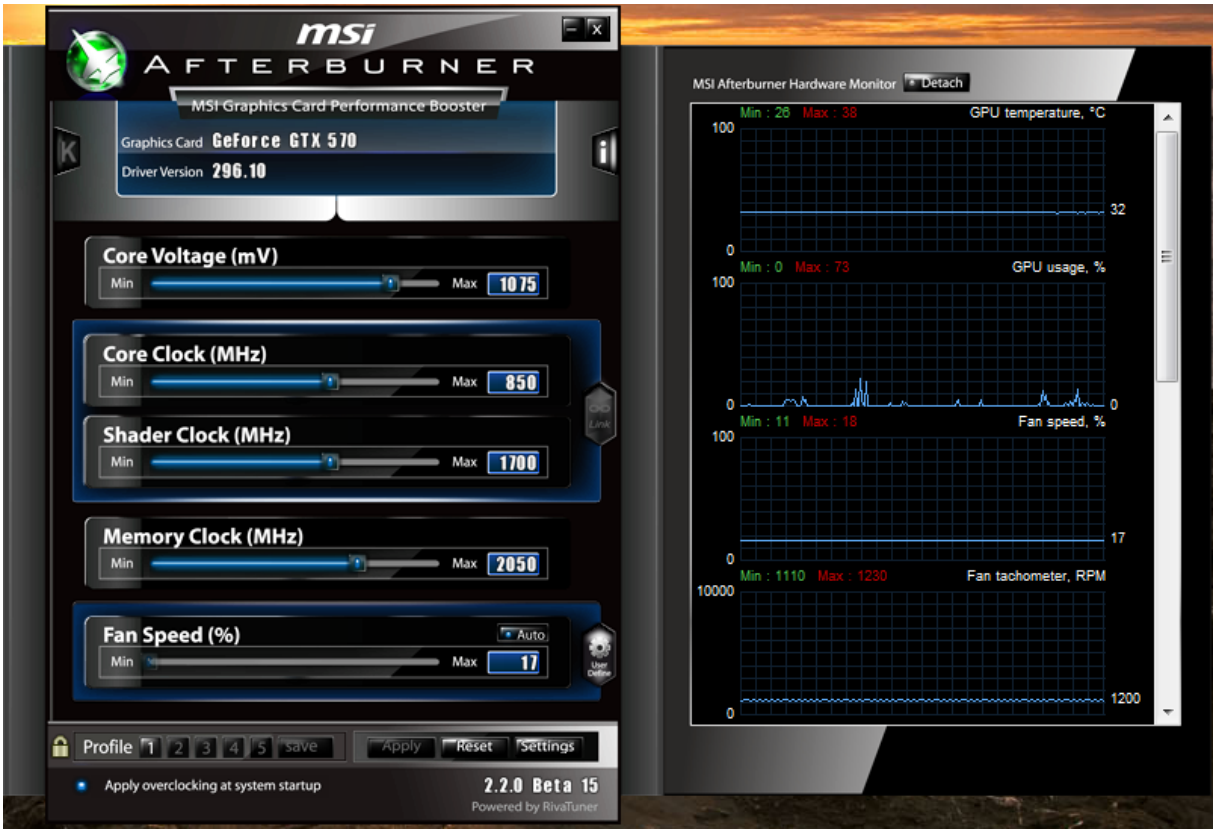
Jedná se o aplikaci k ladění výkonu, která je vytvořena ve spolupráci firmy MSI a právě Rivatuner. Jde už o moderní a velmi detailní nástroj k ladění. Každým týdnem vycházejí aktualizace, aby uspokojili všechny požadavky uživatelů a řešili podporu pro nově vydané grafické karty. Navíc MSI Afterburner zvládá nastavování i monitoring.

3.2.3.1 Co všechno jde ladit

Jde měnit frekvence jádra, frekvence "Shader"(Slouží k renderování . Bohužel od řady Fermi GTX 5xx už nelze manuálně měnit. Je nastavena na "hotclock", to znamená dvojnásobek frekvence jádra), frekvence pamětí. Nadstandardní nastavení je změna napětí na jádru. Díky zvýšení napětí můžu v testech nastavit i hodně vysokou frekvenci jádra a grafická karta je stále stabilní. Každé nastavení lze měnit od opravdu nízkých a bezpečných hodnot, až po hodnoty nebezpečné, kde už hrozí poškození grafické karty. Takže uživatel musí mít zkušenosti, pokud chce využívat takto mocný nástroj. Poslední z důležitých nastavení je rychlost větráčků osazených na grafické kartě. Lze nastavit manuální mód, který se moc nedoporučuje nebo Auto mód, který mění rychlost v závislosti na teplotě jádra. To jde podle nastavení uložené v biosu. MSI Afterburner navíc dokáže upravit toto nastavení podle interaktivního grafu.

3.2.3.2 Co všechno jde monitorovat

Po spuštění nám ukáže, jaký model grafické karty vlastníme a ovladač, na kterém běžíme. Další informace, které MSI Afterburner monitoruje už jsem ve formě dynamického grafu, které běží na časové ose v reálném čase. Monitoruje tedy teplotu grafického jádra (ve stupních celsia), využití jádra(v procentech), rychlost otáčení větráčků (v otáčkách za minutu), frekvenci jádra (v MHz), frekvenci shaderů (v MHz), frekvenci paměti (v MHz), využití paměti grafiky(v MB) a počet snímků (ve snímcích za sekundu, pokud běží aplikace, u které se překresluje celá scéna)



Obrázek 5. Uživatelské rozhraní MSI Afterburner

Na obrázku 5 si lze všimnout v levé horní části typu grafické karty, kterou vlastníme a ovladače, jež obsluhuje tuto kartu. Níže jsou už možnosti pro úpravu parametrů GPU. Pod nimi existuje možnost pro uložení až pěti různých profilů nastavení. Na pravé části obrázku jsou umístěny grafy, které snímají informace v reálném čase.

3.3 Aplikace k testování

3.3.1 MemtestG80

MemtestG80 je softwarový tester k testování na "soft error" v pamětí nebo logiky pro nVidia CUDA grafické karty. Využívá rozmanité prověřené testovací vzory (obyčejné a některé založené na Memtest86 běžící na CPU) k ověření korektnosti operací GPU paměti a logiky. MemtestG80 je užitečná aplikace k ověření grafické karty, že neprodukuje "silent errors", které mohou poškodit výsledky při výpočtech.

3.3.2 FLACCL

FLACCL neboli FlaCuda je jeden s prvních opensource CUDA programů na převod zvukového formátu MP3 (koncovka .wav) do bezeztrátového formátu FLAC(.flac) FLAC je otevřený bezeztrátový, zvukový kodek. Ztrátový formát jako MP3 má některé nedůležité informace vypuštěny (ty co nejsou důležité pro poslech, nejsou slyšet), ale FLAC nevypouští žádnou informaci. Nevýhoda je, že soubory .FLAC maje nepoměrně větší velikost.

Aplikace se spustí příkazem:

CUETools.FLACCL.cmd.exe -10 vstupniSoubor.wav -o vystupniSoubor.flac

Parametr *-10* je kompresní level, *vstupniSoubor.wav* je 250 MB velky zvukový soubor. výstupníSoubor.wav bude název výstupního souboru i s jeho cestou umístění.

3.3.3 LoiLoScope2

První z aplikací s grafickým rozhraním mnou testovaných. Jedná se o nástroj pro práci s videem, který je určen pro začínající nadšence práce s videem. Všechny funkce jsou řádně vysvětleny v grafickém návodu. Obsahuje funkce, které využívají nVidia CUDA. Jedná se o přehrávání videa v průběhu úprav videa. Ale hlavně dokáže projekt převést do formátu H.264 (nebo také mp4) pomocí CUDA.

3.3.4 Extreme GPU Bruteforcer

Jedná se o první aplikaci sloužící k zjištění hesla z Hash, který využívá plný potencionál CUDA a běží tedy pouze na grafických kartách nVidia. Další z aplikací spustitelných pouze v příkazovém řádku. Poradí si se spoustou hashovacích funkcí. Příkladem jsou MD4, MD5, SHA-512, SHA-256, NTLM. V současnosti nejrozšířenější funkcí je MD5, která se ve větší míře využívá k ukládání hesel. MD5 je kryptografická hashovací funkce, která ze vstupu vytvoří výstup s fixní délkou. I malá změna vstupu vytvoří velmi odlišný výstup.

Aplikace se spustí příkazem: *MD5.exe [Soubor s nastavením] <Hash soubor>* .

Kde *Soubor s nastavením* je předem nastavený soubor, kde lze upravovat nastavení a *Hash soubor* je textový soubor s hash kódem.

4. Metodiky testování

4.1 Metodika škálovatelnosti GPU

Hlavní myšlenkou je měření závislosti frekvence jádra grafické karty na frekvenci grafické paměti. Měření vždy probíhají na těchto frekvencích jádra (405 MHz, 500 MHz, 600 MHz, 700 MHz, 800 MHz, 900 MHz) a frekvenci grafické paměti (950 MHz, 1200 MHz, 1400 MHz, 1600 MHz, 1800 MHz, 2000 MHz). Tyto hodnoty jsem zvolil podle sebe a vždy se jedná o nejmenší hodnotu, která jde zadat a o nejvyšší hodnotu, kterou jsem ještě uznal za bezpečnou, aby nedošlo ke zničení celé grafické karty.

4.1.1 MemtestG80

Po spuštění se aplikace zeptá, pokud chceme shromáždit informace o naší grafické kartě. Pro jednodušší měření jsem zvolil možnost "ne". Test se pustil v defaultním módu, což znamená, že běží na první grafické kartě (pokud mám v PC více grafických karet v SLi), jinak samozřejmě běží na jediné grafické kartě. Dále test využívá 128MB RAM a proběhne 50 iterací tohoto testu. Tohle zopakuji pro každou výše uvedenou frekvenci a odečítám časovou složitost testu. Výsledky jsou přesné, protože je odečítám na časové ose "GPU usage" v ladícím programu MSI Afterburner. Je to interval, kde je vytížení grafické karty viditelné. Výsledky si sepíšu do tabulky viz. Výsledky škálovatelnosti GPU.

4.1.2 FLACCL

Aplikaci spustím s parametry:

CUETools.FLACCL.cmd.exe -10 vstupniSoubor.wav -o vystupniSoubor.flac

Po odentrování se už aplikace rozběhne. Navíc obsahuje funkci pro výpočet délky převodu do FLAC formátu. Tyto údaje tedy sepisuju do tabulky.

4.1.3 LoiLoScope2

V aplikaci jsem vytvořil nový projekt. Přidal jsem do něj jeden videozáznam a jeden audio záznam. Tyto dvě stopy jsem sloučil do sebe. Ořezal je na délku 10minut 40sekund. Nakonec jsem projekt převedl do formátu mp4, který využívá právě výpočtů CUDA. Časovou složitost výpočtů jsem vyčetl z grafu na časové ose, využití grafického jádra "GPU usage" v ladícím programu MSI Afterburner a následně sepsal do tabulky.

4.1.4 Extreme GPU Bruteforcer

Pro účel tohoto testováni je potřeba vytvořit heslo. Já jsem zvolil *lexik47*. Nyní jej převedu pomocí MD5 generátoru do hash. Výsledkem je *eda151109c2dbb1f7fd117efeaee334b.* Tento Hash je potřeba uložit do textového souboru, v mém případě *hash.txt.*

Aplikaci spustím příkazem: *MD5.exe MD5.ini hash.txt*

Měřím čas výpočtu pomocí vnitřní funkce Extreme GPU Bruteforcer, která po dokončení výpočtu vypíše délku trvání testu.

4.2 Metodika měření chybovosti

4.2.1 Závislost napětí GPU na frekvenci jádra

K tomuto testu jsem využil MemtestG80 k měření chybovosti při výpočtech, MSI Afterburner k nastavování napětí na GPU, frekvence jádra a monitorování teploty a AIDA64 Extreme (pomocí algoritmu využívající CUDA) jako zátěžový test. Měření jsem prováděl v rozmezí 950 mV až 1025 mV krokem 25 mV a 405 MHz až 900 MHz krokem 100 MHz na grafickém jádru. Test jsem prováděl na základním nastavení. Vždy jsem v MSI Afterburner nastavil patřičné hodnoty a odečítal chybovost z výsledků uvedených v MemtestG80. Teplotu na jádře jsem udržoval na konstantní teplotě 50^{o}C, aby nebyly ovlivněny výsledky tímto faktorem, který je předmětem následujícího měření. Udržování teploty vyžadovalo manuální spouštění a vypínání zátěžového testu na dané teplotě.

4.2.2 Závislost frekvence GPU na teplotě

Opět jsem využil k otestování aplikaci MemtestG80 k odečítání počtu chyb a k ladění výkonu GPU program MSI Afterburner. Pro potřeby testu bylo potřeba využít navíc program AIDA64 Extreme, kterým jsem simuloval vytížení GPU a tak zvyšoval teplotu na hodnotu, kterou jsem potřeboval. Teplotu bylo navíc potřeba regulovat pomocí regulace otáček větráků na grafické kartě. Oblast měření jsem stanovil v rozmezí frekvencí 700 MHz až 900 MHz a teplotě 50^{o}C až 85^{o}C na GPU. MemtestG80 bylo potřeba spouštět v přesně daný čas. Napětí na jádru grafické karty bylo nastaveno na hodnotu 1025 mV. Jedná se o mírně zvýšenou referenční hodnotu, aby nebyly výsledky zkresleny tímto faktorem.

4.3 Metodika měření spotřeby

4.3.1 Metodika měření spotřeby sestavy

K testu jsem použil ladící program MSI Afterburner, aplikaci AIDA64 Extreme k nasimulování zátěže grafické karty a Wattmetr, který měří odběr elektrické energie počítačové sestavy. Test probíhal následovně. Na PC jsem spustil zátěžový test pomocí AIDA64 Extreme, který běžel celou dobu testu. V MSI Afterburneru jsem nastavoval potřebné hodnoty napětí a frekvence jádra. Výsledky jsem odečítal z Wattmetru připojeného k PC.

4.3.2 Efektivita zvýšení napětí GPU

Využiju naměřená data z tabulky 11 a vytvořím tabulku novou, která bude obsahovat frekvenci jádra a spotřebu počítačové sestavy při nejnižším možném napětí na jádru GPU.

5. Výsledky měření

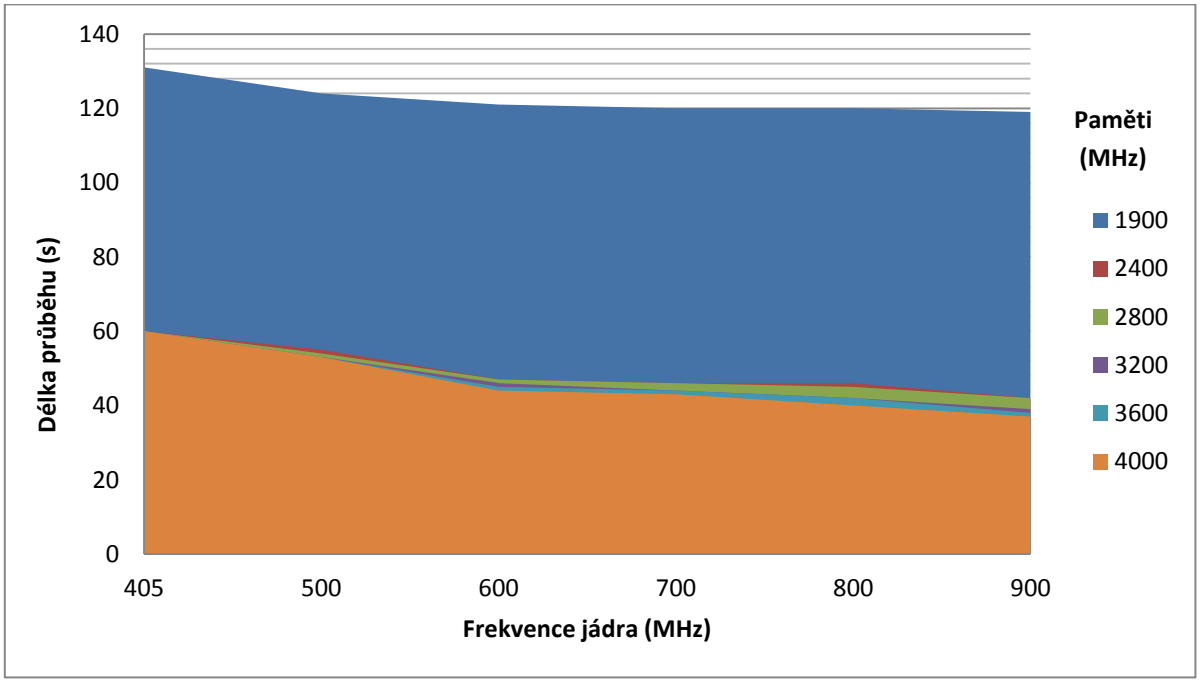
5.1 Výsledky škálovatelnosti GPU

5.1.1 MemtestG80

Paměti/Jádro (MHz)	405	500	600	700	800	900
1900	131	124	121	120	120	119
2400	60	55	47	46	46	42
2800	60	54	47	46	45	42
3200	60	53	46	44	42	39
3600	60	53	45	44	42	38
4000	60	53	44	43	40	37

Měřím délku průběhu jednoho testu. Uvedené výsledky v tabulce jsou v sekundách.

Tabulka 5: Naměřené hodnoty MemtestG80



Obrázek 6: 3D graf měření MemtestG80

Komentář

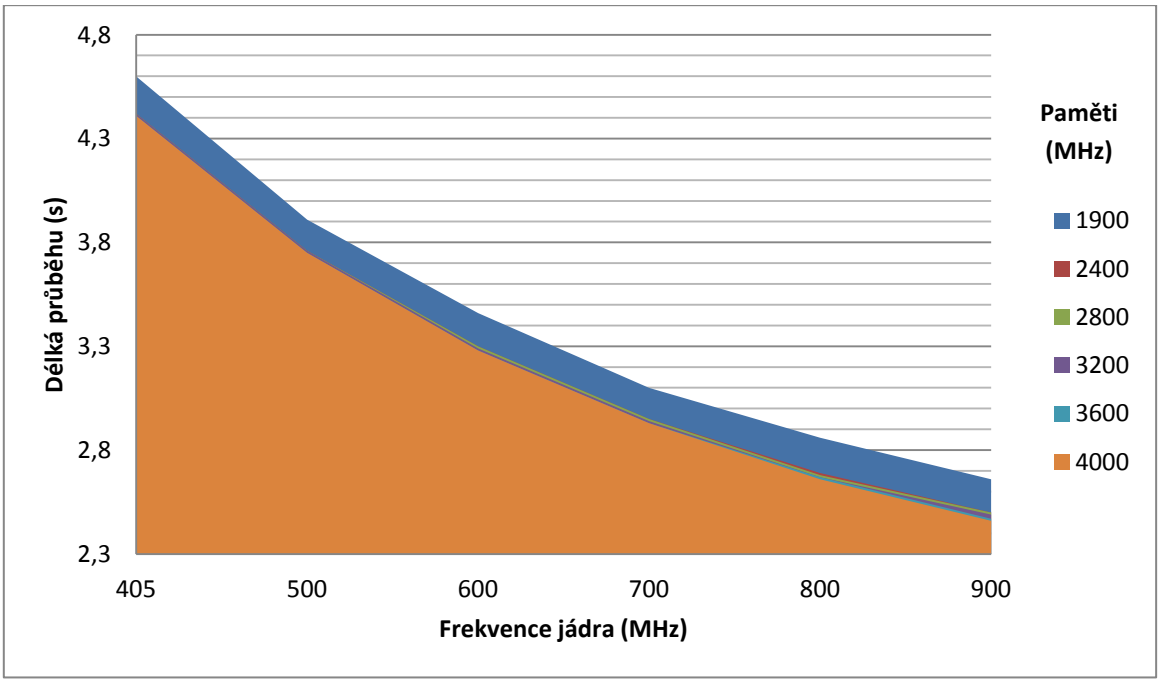
Test MemtestG80 probíhal od 37 sekund až po 131 sekund. Jádro grafické karty bylo vytíženo na 95% u nejnižší hodnoty frekvence na GPU a naopak o poznání nižší vytížení probíhalo na vyšších taktech GPU, kde se pohybovalo okolo 65%. Teploty však nějak výrazně nestoupaly a pohybovaly se v rozmezí 46 až 55^{O}C. Z tabulky a grafu lze vyčíst, že tato aplikace není výrazně závislá na frekvenci paměti. Rozdíly se pohybovaly v jednotkách sekund. Pouze na frekvenci paměti 1900 MHz se objevila velmi výrazná limitace a průběh testu se prodloužil. Frekvence jádra měla výraznější dopad na výkon aplikace a průběh se urychlil při jeho zvýšením hodnoty.

5.1.2 FLACCL

Paměti/Jádro (MHz)	405	500	600	700	800	900
1900	4,6	3,91	3,46	3,1	2,86	2,66
2400	4,42	3,76	3,3	2,95	2,69	2,5
2800	4,42	3,76	3,3	2,95	2,68	2,5
3200	4,42	3,76	3,29	2,94	2,67	2,49
3600	4,41	3,75	3.28	2,93	2,67	2,47
4000	4,41	3,75	3,28	2,93	2,66	2,46

Výsledkem měření je časová náročnost převodu do formátu FLAC a je udán v sekundách.

Tabulka 6: Naměřené hodnoty FLACCL



Obrázek 7: 3D graf měření FLACCL

Komentář

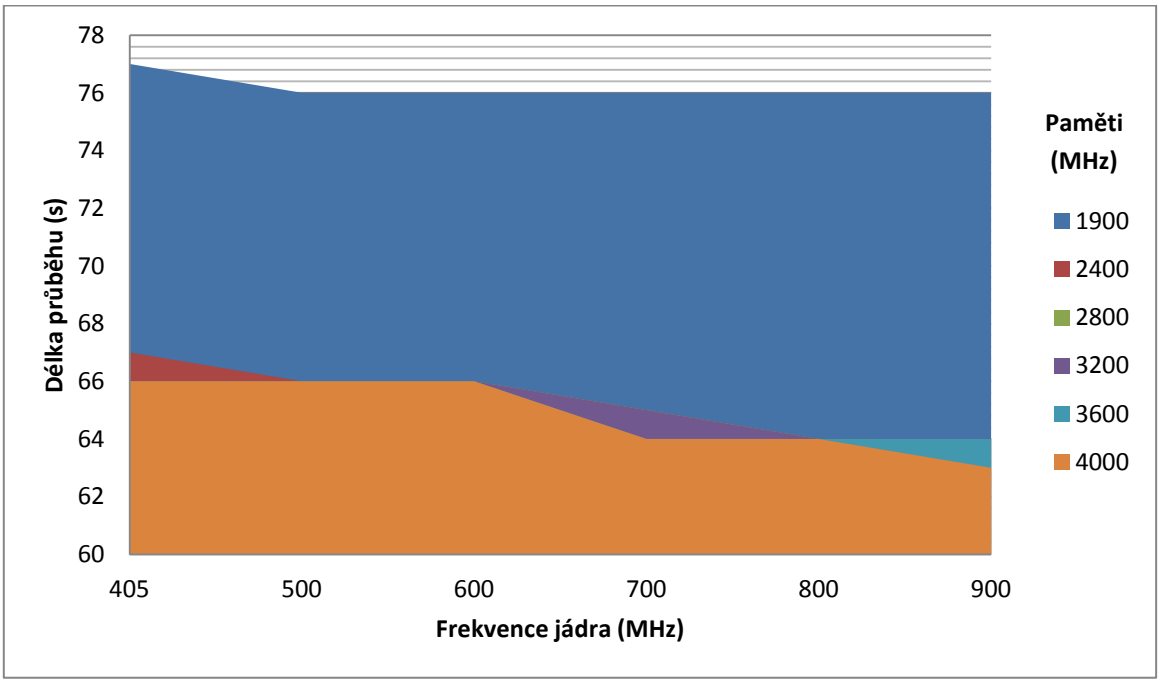
Aplikace FLACCL ukázala potencionál CUDA aplikace a 250MB zvukový soubor převedla do formátu FLAC v jednotkách sekund. Aplikace se ukázala jako silně závislá na frekvenci jádra a při zvyšování hodnot se výkon aplikace mohutně zvyšoval, až na hodnotu 2,46 s. Minimální závislost je však na frekvenci pamětí, kde se výsledky zlepšovaly jen velice mírně, ale opět (jak se stalo v předchozím testu) jsem zjistil limitaci, pokud byly paměti nastaveny na 1900 MHz. Vytížení GPU bylo v rozmezí 85% u nízkých taktů, až po 74% u vyšších frekvencí. Teploty se na GPU nijak nezvýšil hlavně z důvodu velice krátké doby výpočtu i velkých zvukových souborů.

5.1.3 LoiLoScope 2

Paměti/Jádro (MHz)	405	500	600	700	800	900
1900	77	76	76	76	76	76
2400	67	66	66	65	64	64
2800	66	66	66	65	64	64
3200	66	66	66	65	64	64
3600	66	66	66	64	64	64
4000	66	66	66	64	64	63

Výsledkem měření je trvání převodu mnou vytvořeného videosouboru do formátu mp4. Naměřené hodnoty jsou v sekundách.

Tabulka 7: Naměřené hodnoty LoiLoScope 2



Obrázek 8: 3D graf měření LoiLoScope 2

Komentář

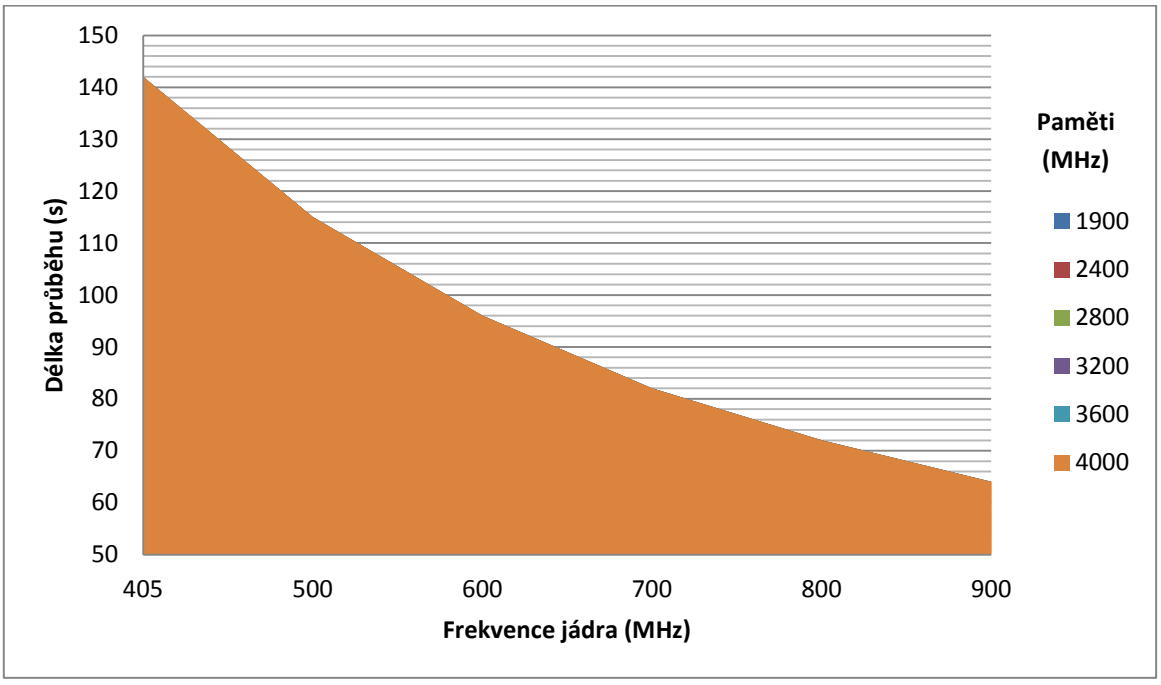
Výsledky LoiLoScope 2 mě docela zklamaly. Zase se ukázala že nejnižší měřená frekvence na paměti způsobovala limitaci, ale při dalším zvyšování frekvence se výkon absolutně nepohnul. Dalším překvapivým faktem je i téměř žádné zvýšení výkonu pomocí zvyšování frekvence na GPU. V celém rozsahu měření (vynechávám limitaci způsobenou grafickou kartou) se délka převodu pohybovala mezi 67 s až 63 s. LoiLoScope 2 při převodu formátu využívá velice málo možnosti grafické karty. Při převodu bylo GPU využívano na 45% (u nízkých frekvencí pamětí a jádra) až po velmi nízkou hodnotu 20% (u nejvyšších frekvencí pamětí a jádra). Z toho vyplývá i málo odpadního tepla a teplota GPU při výpočtu nepřesáhla 48°C.

5.1.4 Extreme GPU Bruteforcer

Paměti/Jádro (MHz)	405	500	600	700	800	900
1900	142	115	96	82	72	64
2400	142	115	96	82	72	64
2800	142	115	96	82	72	64
3200	142	115	96	82	72	64
3600	142	115	96	82	72	64
4000	142	115	96	82	72	64

V tabulce jsou uvedeny výsledky časové náročnosti prolomení hesla z MD5 hash. Jsou sepsány v sekundách.

Tabulka 8: Naměřené hodnoty Extreme GPU Bruteforcer



Obrázek 9: 3D graf měření Extreme GPU Bruteforcer

Komentář

Testovaná kryptografická aplikace Extreme GPU Bruteforcer je příkladem aplikace, která využívá možnosti GPU a CUDA naplno. Program se ukázal jako silně závislý na frekvenci jádra. Nárůst výkonu je víc jak 100%. Na frekvenci jádra 405 MHz prolomení hesla trvalo 142 s a na 900 MHz pouze 64 s. Není zde žádná limitace ze strany pamětí a na její frekvenci vůbec nezáleží. Vytížení GPU po celou dobu testu bylo 100%. Tomu odpovídaly i dosažené teploty na GPU. Ty se pohybovaly na hodnotě 75°C. Důvodem je vykonávání velkého množství operací nad relativně malým množstvím dat (všechna potřebná data jsou nahrána v registrech).

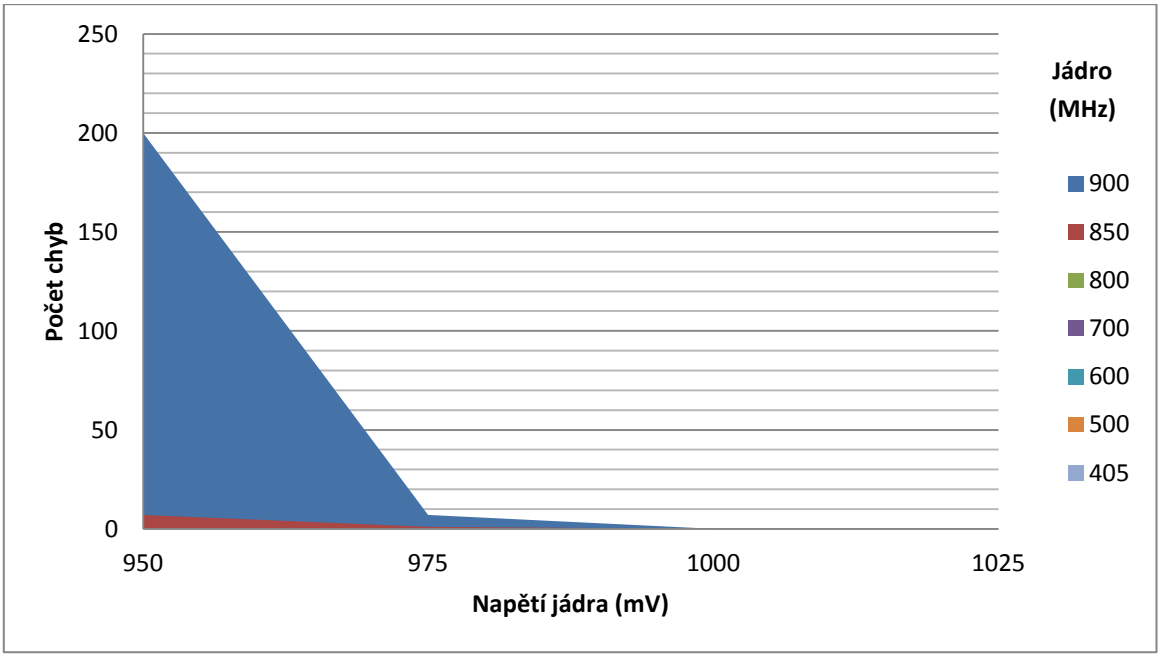
5.2 Výsledky chybovosti

5.2.1 Výsledky závislosti napětí GPU na frekvenci jádra

Jádro (MHz)/Napětí (mV)	950	975	1000	1025
405	0	0	0	0
500	0	0	0	0
600	0	0	0	0
700	0	0	0	0
800	0	0	0	0
850	7	1	0	0
900	200	54	0	0

Výsledkem je počet chyb provedených po čas běhu testu.

Tabulka 9: Naměřené hodnoty chybovosti v MemtestG80



Obrázek 10: 3D graf měření chybovosti

Komentář

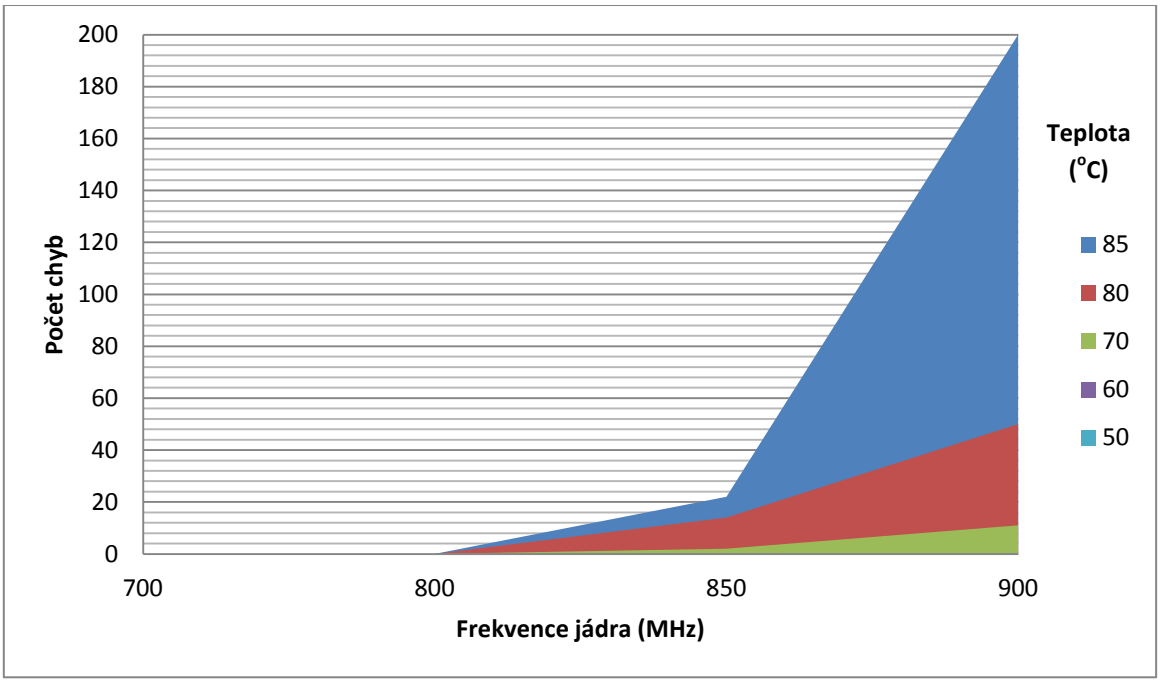
Měření chybovosti výpočtů pomocí MemtestG80 ukázalo zajímavé výsledky, které jsou uvedeny v předchozí tabulce. Nezajímavých výsledků jsem dosáhl pouze při napětí 1000 mV a 1025 mV kde nedocházelo k žádným chybám. Na nižších hodnotách napětí (které však jsou už nižší než referenční) se na frekvencích jádra začaly vyskytovat první chyby ve výpočtech. Konkrétně na frekvencích 850 MHz. Při napětí 950 mV a 900 MHz na GPU již docházelo k nestabilitě systému nebo vysokému počtu chyb.

5.2.2 Výsledky závislosti frekvence GPU na teplotě

Teplota (°C)/Jádro (MHz)	900	850	800	700
50	0	0	0	0
60	1	0	0	0
70	11	2	0	0
80	50	14	0	0
85	X	22	0	0

Výsledkem měření je počet chyb výpočtů, které proběhnou v průběhu testu.

Tabulka 10: Naměřené hodnoty chybovosti v MemtestG80



Obrázek 11: 3D graf měření chybovosti

Komentář

Tabulka výše ukazuje zjištěná fakta. S frekvencí 700 MHz až 800 MHz si jádro grafické karty poradilo bez potíží při jakékoliv měřené teplotě. Avšak při dalším zvýšení frekvence GPU se ukázala teplota jako rozhodující faktor. Na frekvenci 850 MHz začala grafická karta už při teplotě 70°C chybovat. Další zvyšování teploty tento fakt jen dokazoval. Při zvýšení frekvence na hodnotu 900 MHz se chybovost ukázala na nižší teplotě. Navíc při teplotách nad 80°C začal být systém nestabilní a pokud se test podařilo spustit, proběhlo spoustu chyb.

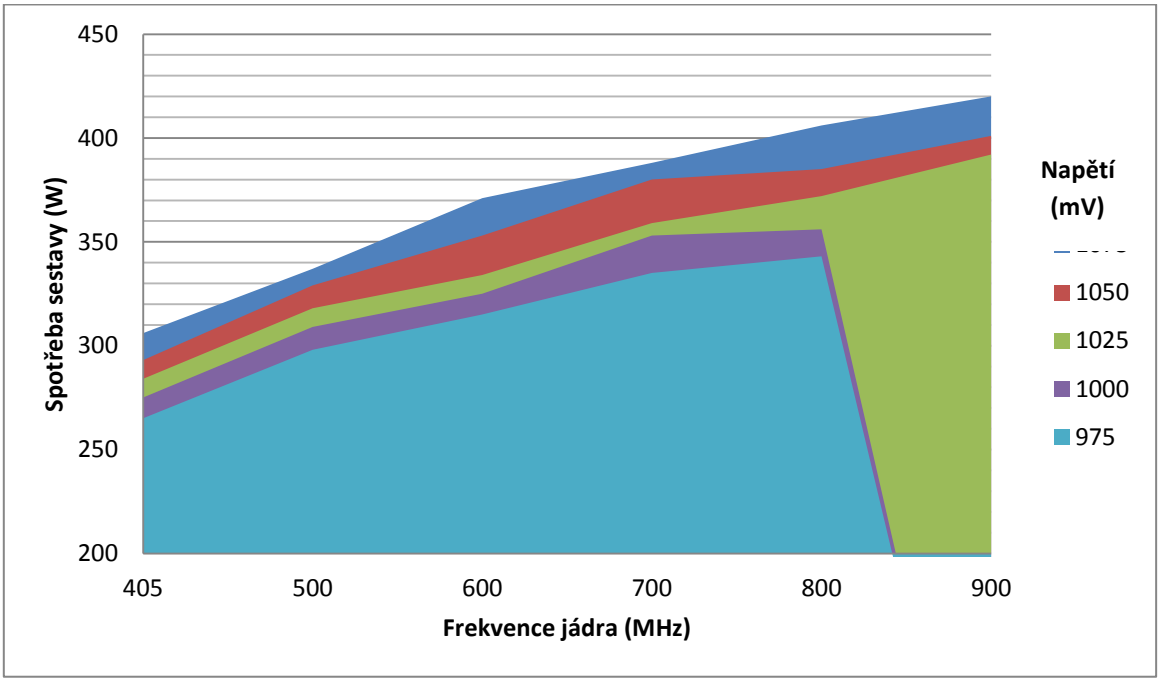
5.3 Měření spotřeby

5.3.1 Výsledky měření spotřeby

Napětí (mV)/ Jádro (MHz)	405	500	600	700	800	900
975	265	298	315	335	343	X
1000	275	309	325	353	356	X
1025	284	318	334	359	372	392
1050	293	329	353	380	385	401
1075	306	337	371	388	406	420

Výsledkem měření je spotřeba celé sestavy ve Wattech (W). Hodnoty X jsou místa, kde sestava už nebyla stabilní a nešlo měřit.

Tabulka 11: Naměřená spotřeba sestavy



Obrázek 12: 3D graf naměřené spotřeby sestavy

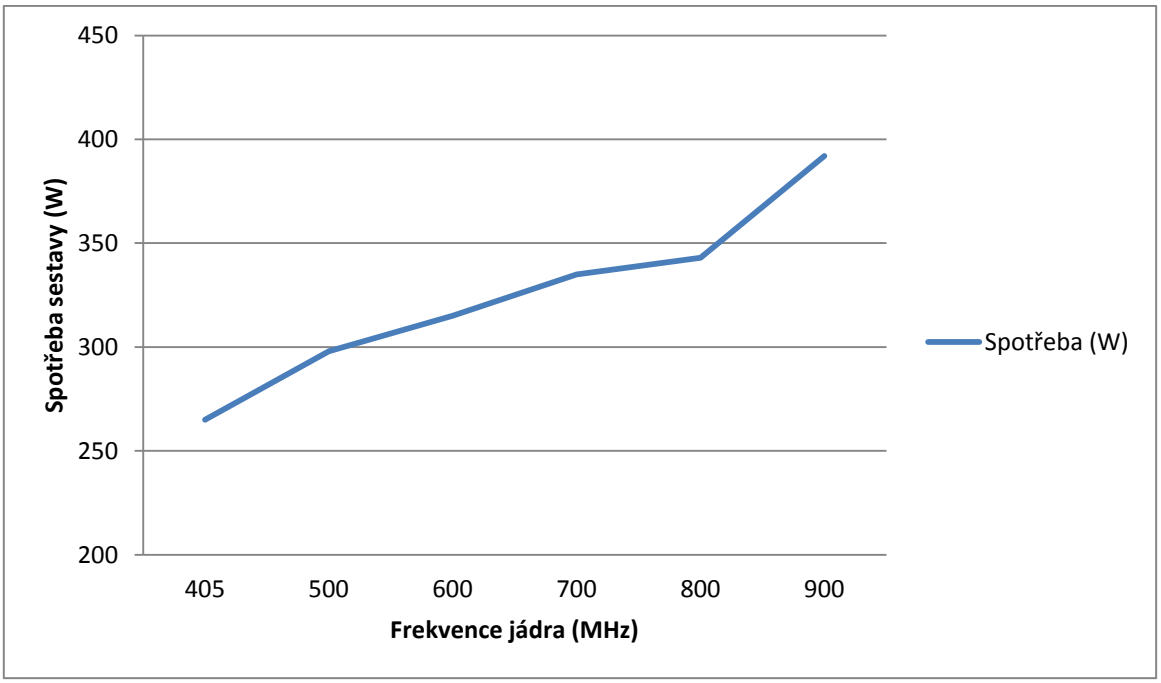
Komentář

Test měření spotřeby ukázal vysoký nárůst spotřeby i při mírném zvýšení parametrů. Zvyšování napětí o 25 mV na každé frekvenci jádra zvětšilo odběr o 10 až 20 W. Postupné zvyšování frekvence o 100Mhz zase zvětšovalo spotřebu o 20 až 30 W. Význam zvyšovat napětí na GPU se ukázal na frekvenci 900 MHz, kde jde vidět, že pokud nebylo přiváděno do jádra 1025 mV, nebyl systém stabilní. Teploty GPU se vyšplhaly až na 89 °C pokud jsem přiváděl do jádra napětí 1075 mV. Otáčky chladiče v tomto případě pracovaly na 50%, což už je velice hlučný stav. Spotřeba sestavy se celkově pohybovala od 265 do 420 W. Oproti spotřebě počítačové sestavy v klidu, kde se pohyboval odběr elektrické energie na 90 W je to výrazné zvýšení.

5.3.2 Výsledky efektivity zvýšení napětí GPU

Jádro (MHz)	405	500	600	700	800	900
Spotřeba (W)	265	298	315	335	343	392

Tabulka 12: Efektivita zvyšování frekvence jádra



Obrázek 13: Graf efektivity zvyšování frekvence jádra

Komentář

Tabulka výše ukazuje efektivitu zvýšení napětí, tedy jestli se ještě vyplatí zvyšovat napětí, abychom získali větší výkon. Celé měření od 405 MHz až po 800 MHz jsem si vystačil s napětím 975 mV. Ale abych dosáhl stabilního výsledku na frekvenci vyšší, musel jsem zvýšit napětí na GPU o 50 mV. Takže tím že jsem dosáhl stabilního stavu na frekvenci 900 MHz, vzrostla spotřeba o 49 W. Už se jedná o velké zvýšení spotřeby, ale tímto krokem jsem získal víc jak 10% výkonu navíc.

6. Závěr

Tématem této bakalářské práce bylo ladění výkonu grafické karty v aplikacích CUDA. Nejprve bylo potřeba vysvětlit problematiku GPGPU obecně, kde jsem nastínil, co to GPGPU je, oblast možného využití a metody které se využívají u výpočtů. Bylo potřeba oddělit hardwarové (dále jen HW) a softwarové prostředky (dále jen SW) GPGPU. V podkapitole HW prostředků jsem se zaměřil na programovatelné jednotky v GPU. Ukázal jsem vývoj v této oblasti a uvedl třídy Shader modelů a příklady instrukcí. Poté jsem objasnil HW část technologie CUDA. Do podkapitoly SW prostředků jsem zařadil předpoklady pro programování v CUDA, ukázkový program a zmínku o PhysX. Dále jsem zmínil největší konkurenci CUDA, jimiž jsou OpenCL vyvíjené průmyslovým konsorciem Khronos a Microsoft DirectCompute.

Další kapitola je zaměřena na testovanou grafickou kartu a software, který využívám pro ladění výkonu GPU a samotné testovací aplikace. Zvolil jsem GeForce GTX 570 a popsal její obecné vlastnosti a parametry. Dále jsem uvedl problematiku a historii vývoje ladících programů od Rivatuner, až po současné nejvýkonnější programy, kterým bezesporu je MSI Afterburner. Zbývalo jen vypsat aplikace, které jsem využil k testování.

Obsahem následující kapitoly bylo sepsání metodiky pro jednotlivá měření. Provedl jsem testy škálovatelnosti výkonu GPU na 4 různých testovacích aplikacích. Krok po kroku jsem popsal přesný postup měření. Dalším testem bylo zjištění chybovosti výpočtů. Zde jsem počítal s dvěma ovlivňujícími faktory. Vytvořil jsem metodiku pro měření chybovosti v závislosti napětí na frekvenci GPU a metodiku závislosti frekvence na teplotě GPU. Poslední metodika byla pro měření spotřeby počítačové sestavy a následnou efektivitu zvyšování napětí GPU.

Následující kapitola už obsahovala výsledky měření ve formě tabulky a z něj vytvořeného grafu. Kapitola obsahovala i komentář zachycující zajímavá fakta v průběhu měření.

Vývoj v oblasti grafických karet jde kupředu vysokou rychlostí. Nejvýznamnější výrobci grafických karet nVidia a AMD vedou tvrdý konkurenční boj a každý rok uvádí obě společnosti novou řadu karet a co 2 až 3 roky novou architekturu. Vždy se jedná o nárůst okolo 20% při stejné spotřebě GPU. To je dosaženo využitím stále dokonalejších výrobních procesů. V současnosti existuje 22nm výrobní proces. V této oblasti se nedá skutečně co vytknout. Co se týče samotného ladění výkonu, zde jsme závislí na výrobcích grafických karet. Je důležité jak vysoko nastaví parametry GPU (vždy se to děje tak, aby předstihli konkurenci ve výkonu o pár procent). Pokud jedna ze stran vyrobí výborný produkt, je zde možnost na obrovské dodatečné přetaktování (v desítkách procent). V testech jsme dokázal že zvyšovat parametry grafické karty smysl má. Ladící programy díky své oblibě v domácím použití dostávají týdenní aktualizace a okamžitou podporu pro nové produkty.

7. Seznam tabulek a obrázků

8. Použité zdroje

Literatura

[1] Microsoft: DirectX SDK. Online, 2012, [cit. 11. Dubna 2012].
Dostupné na webové stránce:
http://msdn.microsoft.com/en-us/library/windows/desktop/ee663274(v=vs.85).aspx

[2] NVIDIA Corporation. Cuda Threading Model, Online, 2012, [cit. 1.Dubna 2012] Dostupné na webové adrese: http://courses.engr.illinois.edu/ece498/al/textbook/Chapter3-CudaThreadingModel.pdf

[3] NVIDIA Corporation. PhysX by NVIDIA, PhysX SDK Introduction Online, 2012, [cit. 2.Dubna 2012] Dostupné na webové adrese:
ftp://download.nvidia.com/developer/cuda/seminar/TDCI_PhysX.pdf

[4] Khronos: OpenCL. Online, 2012, [cit. 5. Dubna 2012].
Dostupné na webové adrese:
http://kite.khronos.org/assets/uploads/developers/library/overview/opencl_overview.pdf

[5] Microsoft: DirectCompute. Online, 2012, [cit. 4.Dubna 2012]. Dostupné na webová adrese:
http://www.nvidia.com/content/GTC/documents/1015_GTC09.pdf

[6] RivaTuner. Online, 2012, [cit. 11. Dubna 2012]. Dostupné na webové adrese:
http://www.guru3d.com/category/rivatuner/

[7]Stanford: Folding@Home. Online, 2012, [cit. 5.Dubna 2012]. Dostupné na webové adrese:
http://folding.stanford.edu/English/Science

[8] NVIDIA Corporation: NVIDIA CUDA Programming Guide. Online, Santa Clara, 2010.
Dostupné na webové stránce:
http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_Programming Guide.pdf

[9] Owens, J. D.; Luebke, D.; Govindaraju, N.; aj.: A Survey of General-Purpose Computation on Graphics Hardware. In Eurographics 2005, State of the Art Reports, Dublin, Ireland, August 29– September 2 2005 . Dostupné na webové stránce:
http://www.crc.nd.edu/~rich/SC06/sc06_tutorials/S07/S07.tutorial.2up.pdf

9. Příloha

I. Naměřené hodnoty